Dear CROSSTALK Editor,

It is often said today that the difference between software reliability and hardware reliability is that software does not fatigue, wear out, or burn out.

Thirty years ago, it was common to read of computers that would become sluggish and ineffective after running for a while. The simple treatment was to shut down and restart. Eventually it was realized that the problem was caused by poor memory management. The invention of *garbage disposal*, when used, takes care of it. Unfortunately even nowadays good memory management is not found as often as it should be, and Professor Trivedi at Duke University has done a deal of work trying to educate people in the effects of what he calls *aging* and what to do about it.

It could as easily be called *wear*. Consider the following: we speak of brakes wearing when they lose material, of structures wearing by fatigue (change in the crystalline structure), and of lubricating oil wearing out (chemical change). There is no common factor except simply reduced usability as a result of use. And software with badly managed memory suffers reduced usability as a result of use.

It is not a matter of which heated argument is worthwhile; however, unless software designers recognize that they must design to prevent reduced usability as a result of use, there will be systems that have to be frequently restarted, causing a nuisance.

Roderick Rees
Boeing

Dear CROSSTALK Editor,

In her article, "How and Why to Use the Unified Modeling Language" (June 2005 CROSSTALK), Lynn Sanderfer gave a useful survey of the UML and the benefits it can bring to a development process. It is also useful to discuss some of the disadvantages of UML.

The developers must learn from the users enough to make a domain description that is sufficient to create the application. UML is not a satisfactory notation for a domain description, because it is too hard for users to read.

There is no standard for saving and exchanging UML, so there is a risk to maintainability in being locked in to a proprietary tool. XML records the model but not the layout of the diagram, and the XML standard is a long way short of guaranteeing portability. In fact, there is a commercial market in XML conversion from one proprietary dialect to another.

Most tools can export to the XML MetaData Interchange (XMI) format. Unfortunately XMI records the model but not the layout of the diagram, and the XMI standard is a long way short of guaranteeing portability. Not only are its versions very different, but proprietary extensions are common, and the Object Management Group does not provide tests for compliance. (In general, these are the features of UML that cannot be mapped to the Meta-Object Facility.)

Generating code from UML is not easy. There is no single tool or interface that has become standard for doing this. UML is stored in a binary file, so changes to the model by different developers cannot be merged automatically. The collaboration of a team of developers, especially a distributed team, depends on a source code repository and is founded on the diff utility. The use of UML leads to a directive style and a waterfall process, which is not suitable for all projects.

UML provides a good way to visualize object-oriented software. It is a suitable tool for some tasks. But a process in which a UML model is the central artifact is unsuitable for many projects.

Chris Morris
Daresbury Lab
Warrington, UK